

IoT アプリ構築支援のための SINETStream Android ライブラリおよびセンサ情報収集アプリの開発

竹房 あつ子^{1,2,a)} 小林 久美子¹ 北川 直哉¹ 孫 静涛^{1,†1} 吉田 浩¹ 合田 憲人^{1,2}

受付日 2022年8月23日, 採録日 2023年1月19日

概要: 各種センサデータをクラウドに収集, 蓄積し, 機械学習で高度なデータ解析を行う IoT アプリケーションシステムの構築が可能になってきた. スマートフォンは, カメラ, 加速度計, GPS 等の多様センサやバッテリーがあらかじめ装備されており, IoT システムへの活用が期待されるが, プログラム開発コストの高さから特に学術分野では十分に活用されていない. 本研究では, Android スマートフォン向けのアプリ開発を支援するライブラリの開発と, 開発したライブラリを利用した Android センサ端末用アプリを開発し, 研究・教育を目的とした IoT アプリケーションシステムの構築, 利用促進の支援を目指す. 我々は, 広域データ収集・解析プログラム開発支援ソフトウェアパッケージ SINETStream を開発している. 本稿では, Python/Java 版のライブラリに加えて Android 版ライブラリを新たに開発し, その基本性能を示す. また, 開発した Android 版を利用してテキスト送受信アプリとセンサ情報収集アプリ (Sensor) を開発した. Sensor アプリは, IoT アプリケーションを開発する際に, 利用者は新たにプログラミングを行う必要がなくそのまま利用可能である. 研究・教育用途での利用を支援するため, チュートリアルも開発し, IoT アプリケーションシステムの構築を体験できるようにした. さらに, 広域での実証実験により Sensor アプリが IoT システムのセンサとして活用できることを示す.

キーワード: IoT (Internet of Things), Android, ソフトウェアライブラリ, Android アプリ, モバイルネットワーク

Development of SINETStream Android Library and the Sensor Data Collection Apps to Support IoT Application Construction

ATSUKO TAKEFUSA^{1,2,a)} KUMIKO KOBAYASHI¹ NAOYA KITAGAWA¹ JINGTAO SUN^{1,†1} HIROSHI YOSHIDA¹
KENTO AIDA^{1,2}

Received: August 23, 2022, Accepted: January 19, 2023

Abstract: It is now possible to build IoT application systems that collect and store various types of sensor data to a cloud and perform advanced data analysis of the data using machine learning. Smartphones are widely used and equipped with a variety of sensors, such as cameras, accelerometer and GPS, and batteries, and are expected to be used for IoT systems. However, they are not fully utilized, especially in academic fields, due to the difficulty of coding. In this study, we develop a software library to support development of IoT application systems using Android-based smartphones and applications for Android sensor devices using the developed library, aiming to support the construction and promotion of IoT application systems for research and education purposes. We have been developing “SINETStream,” a software package to support development of wide-area data collection and analysis programs. In this paper, we have developed an Android version of the library and show its basic performance. Using the developed Android library, we have also developed Android apps, a text sending/receiving application and a sensor information collection application (Sensor), and their tutorials that allow users to experience the construction and use of an IoT system. We will also demonstrate that the developed app can be used for a sensor device in an IoT system through the experiment in a field.

Keywords: IoT (Internet of Things), Android, software library, Android app, mobile network

¹ 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101-8430, Japan

² 総合研究大学院大学
The Graduate University for Advanced Studies (SOKENDAI),

Hayama, Kanagawa 240-0193, Japan

^{†1} 現在, 日立製作所 研究開発グループ

Presently with Hitachi, Ltd., Research & Development Group

^{a)} takefusa@nii.ac.jp

1. はじめに

広帯域、低遅延、長距離通信を可能にする第5世代移動通信システム「5G」の実用化とセンサ端末の小型化・安価化により、モバイルネットワークを介して各種センサから収集された大量な情報をクラウドに蓄積し、機械学習を用いた高度なデータ解析を可能にするIoT (Internet of Things) アプリケーションシステムの構築が可能になってきた。このようなIoTアプリケーションでは、開発コストや端末自体のコストの観点からRaspberry PiのようなLinuxベースの小型端末に必要なセンサを搭載して利用することが多い。一方で、スマートフォンはカメラ、GPS等の多様なセンサやバッテリーがあらかじめ装備されていること、すでに多くの端末が普及していることから、IoTアプリケーションへの活用が期待されている。

国立情報学研究所 (NII) では、学術情報ネットワークSINETをモバイル網に延伸させたモバイルSINET [1]により、センサ端末をSINETのVPNに直接接続させた安全なIoT実験環境の構築を可能にしている。また、応用研究分野の研究者によるIoTアプリケーションの開発を支援するため、広域データ収集・解析プログラム開発支援ソフトウェアパッケージSINETStream [2], [3]を開発し、Linux環境等で利用可能なPython/Java版ライブラリを公開している。しかしながら、Linuxベースのシステムと比較するとスマートフォンの場合はプログラムの開発コストが高く、それにより特に学術分野では十分に活用されていない。

本研究では、SINETStream Android版ライブラリを開発 [4] と、本ライブラリを用いたアプリ [5]を開発し、研究・教育を目的としたIoTアプリケーションシステムの構築、利用の促進を目指す。SINETStream Android版ライブラリは、IoTデータの送受信のための機能を提供するコアライブラリと、Android端末のセンサ情報の収集を支援するヘルパーライブラリで構成される。コアライブラリは、SINETStreamのPython/Java版と同等の機能を提供する。ヘルパーライブラリは、Android端末に装備されたセンサの利用を支援するためのライブラリを提供する。本稿では、その基本性能も示す。

また、Android版ライブラリを利用して、Android用のテキスト送受信アプリSINETStream Android Echo (Echoと呼ぶ)とセンサ情報収集アプリSINETStream Android Sensor Publisher (Sensorと呼ぶ)を開発する。Echoアプリでは、IoTシステムのデータ収集における基本的な振る舞いを確認することができる。Sensorアプリでは、Android端末をセンサとするIoTアプリケーションでのセンサプログラム開発が不要となり、IoTを活用した研究開発を加速させることができる。また、研究・教育用途での利用を支援するため、EchoとSensorを用いたチュートリア

ルも開発、公開し、IoTアプリケーションシステムの構築、利用を簡単に体験できるようにする。最後に、フィールドでの実証実験としてSensorアプリを用いて東京海洋大学の実験航海で船上での複数センサ情報の収集を行った。実験により、Sensorアプリで3日間のセンサ情報収集が欠損なく行えることを確認した。

2. SINETStream の概要

広域データ収集・解析プログラム開発支援ソフトウェアパッケージSINETStreamの概要について説明する。SINETStreamのAPIおよび機能の詳細は、公式サイト [2] および文献 [3] を参照されたい。

2.1 SINETStream API

SINETStreamでは、トピックベースのPub-Sub型非同期メッセージングモデルを前提としている。図1にSINETStreamの概念図を示す。SINETStreamでは、センサデータを送信するIoTデバイス側のPublisherプログラムをWriter、収集したデータを活用するサーバ計算機等で利用するSubscriberのプログラムをReaderと呼び、WriterとReaderでSINETStreamが提供するAPIを実装することで図1の中央にあるメッセージブローカ (ブローカ) を介してセンサデータの送受信を可能にする。ブローカには、SINETStreamに対応した任意のブローカソフトウェア、またはMQTT (Message Queue Telemetry Transport) [6] ベースのクラウドPub-Subサービスを利用することができる。プログラムの可搬性を高めるため、各メッセージブローカとの通信に必要なパラメータセットは別途定義できるようにしている。また、低遅延または高スループットな送受信処理を目的として、同期および非同期のWriter/Reader APIを提供している。

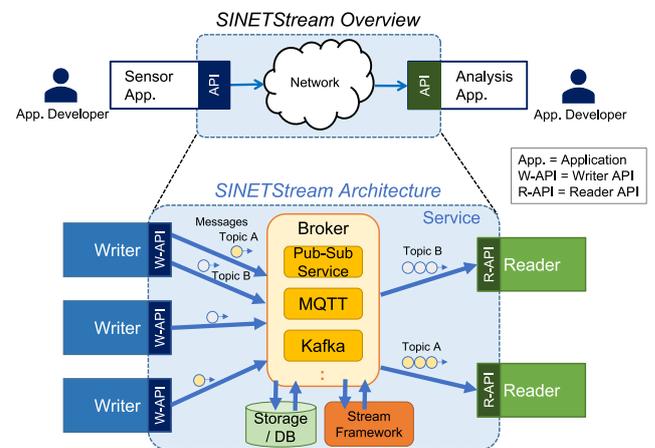


図1 SINETStream の概念図

Fig. 1 Overview of SINETStream.

2.2 SINETStream の機能

SINETStream では、メッセージ送受信のための共通 API の提供のほか、IoT アプリケーションシステムに必要なセキュリティ機能、メトリクス収集機能と、多様なブローカに対応するための SPI (Service Provider Interface) を提供している。図 2 に SINETStream のソフトウェア構成を示す。図 2 上部の青枠内に示した部分は SINETStream コアライブラリであり、API および SPI と、セキュリティ機能など IoT アプリケーションに必要とされる共通機能とプラグイン管理機能が実装されている。

セキュリティ機能として、認証・認可と通信・データの暗号化機能を提供している。認証・認可は、各ブローカ実装でサポートしている認証・認可機能を、SINETStream のコンフィグファイルまたは API から統一的に設定できるようにしている。公開鍵認証またはパスワード認証により、クライアントおよびサーバの認証が可能である。また、TLS による通信時の暗号化と SINETStream で提供しているデータ暗号化機能が利用できる。通信の暗号化は通信時のみ暗号化されるのに対し、データ暗号化はデータ収集サーバ側でも暗号化された状態でデータが格納されることになるため、より安全なデータ管理が可能になる。

メトリクス収集機能では、通信負荷をかけずにメッセージ送受信に関するメトリクス情報を収集することができる。SINETStream ライブラリ内で送受信メッセージ数、メッセージサイズ、エラー数をカウントすることで、ある期間における通信レート、最小/平均/最大メッセージサイズ、エラーレート等を提供することができる。利用者による測定用のデータ送信が必要ないため、通信負荷をかけずにメトリクス収集が可能になる。

また、多様なメッセージブローカに対応できるようにするため、SPI を提供している。図 2 下部の緑枠内で示した部分は各ブローカ用のプラグインを表しており、SPI を介して独立したプラグインモジュールを組み合わせることができる。SINETStream v1.5 時点での Python/Java の実装では、MQTT ベースブローカと Apache Kafka [7], [8]

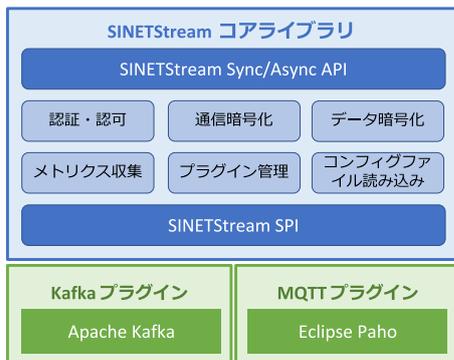


図 2 SINETStream のソフトウェア構成の概要
Fig. 2 An overview of SINETStream modules.

(Kafka) をサポートしている。SPI を実装したプラグインを追加することで、エッジコンピューティング用のメッセージング基盤ソフトウェアなど、最新のメッセージブローカにも対応することができる。

3. SINETStream Android 版ライブラリの開発

Android 版ライブラリでは、SINETStream Python/Java 版で提供している IoT アプリケーションのためのデータ送受信機能のほか、Android デバイ스에 装備されたセンサ情報の収集を容易にするための機能も開発した。図 3 に SINETStream Android 版のソフトウェア構成を示す。図 3 右側のデータ送受信機能を提供するコアライブラリと、左側のセンサ情報収集を支援するヘルパーライブラリで構成されている。図中の矢印はデータの流れを表している。コアライブラリでは、下方向の矢印が Writer が publish する場合のデータの流れ、上方向の矢印が Reader が subscribe している場合のデータの流れを表す。ヘルパーライブラリでは、Android システムから各種センサデータを受け取り、適宜加工してユーザアプリケーションプログラムに送信する。以下でコアライブラリとヘルパーライブラリについて説明し、その後コアライブラリを用いた Android 版ライブラリの基本性能を示す。

3.1 SINETStream Android コアライブラリ

SINETStream Android コアライブラリは、SINETStream Python/Java 版と同等の機能を提供することを目的としている。ただし、Android はスマートフォンでの利用を前提とした OS であり、Android 用の Java では一部機能が制限されている。具体的には、Android 用の Java の機能制限により、非同期呼び出しとコールバックで実装する必要がある。また、これにより Apache Kafka では

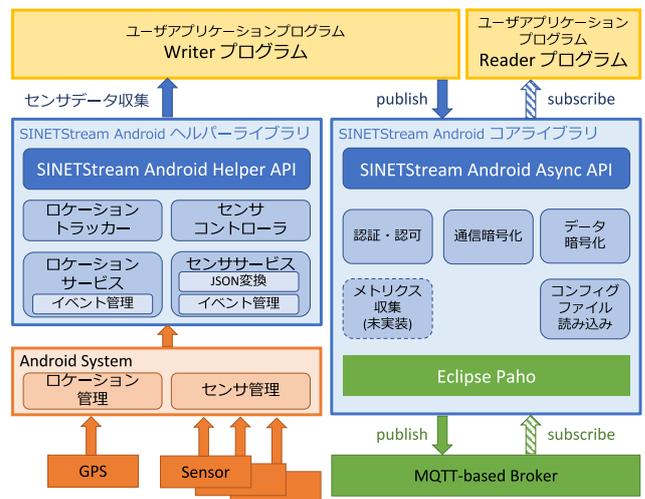


図 3 SINETStream Android 版のソフトウェア構成の概要
Fig. 3 An overview of SINETStream Android modules.

Android 用クライアントライブラリは提供されていない。よって、Android コアライブラリでは Python/Java 版で提供している機能と提供していない機能がある。図 3 右側で示すように、同期 Writer/Reader API は提供していない、SPI およびプラグインは非対応でプラグイン管理機能がなく Eclipse Paho [9] を用いた MQTT ブローカへの通信のみ対応している点が異なっている。また、メトリクス収集機能は今後実装を検討している。

3.2 SINETStream Android ヘルパーライブラリ

ヘルパーライブラリは SINETStream Android 版固有の機能であり、Android 端末に装備された多様なセンサのデータ収集、加工を支援する。ただし、カメラ（動画、静止画）とマイク（音声）はバックグラウンドで継続的に動かし続けることができないため、本ライブラリの対象とはしなかった。図 3 左側に、ヘルパーライブラリのソフトウェア構成の概要と Android システムのセンサ管理およびユーザプログラムとの関係を示す。ヘルパーライブラリは、コアライブラリ同様にユーザプログラムに対して非同期呼び出しとコールバックからなる API を提供し、Android システムのセンサを容易に扱えるようにする。

ヘルパーライブラリは、センサコントローラとセンササービス、ロケーショントラッカーとロケーションサービスの 4 つのモジュールで構成されている。センサコントローラは、Android 端末のセンサを制御するセンササービスのフロントエンドとして機能し、利用可能なセンサタイプの一覧の提供、センサデータの送出間隔の設定等を可能にする。センササービスは、Android 端末が装備する各種センサからセンサデータを受け取って蓄積し、設定された送出間隔でセンサデータをサーバ側で処理しやすいように JSON 形式に加工して送出する。ロケーショントラッカーは、GPS で Android 端末位置を取得するロケーションサービスのフロントエンドとして機能する。ロケーションサービスが収集した位置情報をユーザアプリケーションプログラムに送信する。

図 4 に JSON 形式に加工されたセンサデータの例を示す。加工されたセンサデータでは、各センサで取得されたセンサデータに加え、Android 端末の情報が付与される。ヘルパーライブラリにより、複数センサのデータをまとめて 1 つのメッセージとして送信できるようになる。図 4 では、図 3 左下の Android System の「センサ管理」で収集されるセンサデータを sensors 以下にリスト化している。その他の Android システム情報 sysinfo、ユーザが定義可能なユーザ情報 userinfo、Android System の「ロケーション管理」で取得する位置情報 location を、デバイス情報 device としてまとめた。

```
{
  "device": {
    "sysinfo": {
      "android": "11",
      "model": "Pixel 4",
      "manufacturer": "Google"
    },
    "userinfo": {
      "publisher": "user1@example.com",
      "note": "room1, bldg-A"
    },
    "location": {
      "latitude": "139.xxxxxx",
      "longitude": "35.xxxxxx"
    }
  },
  "sensors": [
    {
      "timestamp": "20200521T171528.094+0900",
      "type": "accelerometer",
      "name": "LSM6DSR Accelerometer",
      "values": [
        -4.1984100341796875,
        1.2285531759262085,
        5.224560737609863
      ]
    },
    {
      "timestamp": "20200521T171528.142+0900",
      "type": "light",
      "name": "TMD3702V Ambient Light Sensor",
      "value": 67.49675750732422
    },
    ...
  ]
}
```

図 4 JSON 形式に加工されたセンサデータ。device はデバイス情報、sensors はセンサデータ情報を表す

Fig. 4 JSON-formatted sensor data. device and sensors indicate device information and sensor data, respectively.

3.3 SINETStream Android の基本性能

SINETStream Android 版ライブラリの基本性能を示すため、コアライブラリを用いて TLS およびデータ暗号化の有無での性能をモバイル環境および Wi-Fi (LAN) 環境で調査する。ただし、モバイル環境は利用可能帯域の変動が大きいため、変動の少ない夜間に測定を行った。

図 5 に実験概要を示す。実験では、NII にある Android 端末で動作する Writer からクラウド計算ノード上で動作するブローカへメッセージを送信し、Android 端末の Reader でブローカに到着したデータの読み込みが完了するまでの時間を測定し、通信スループット、遅延、メッセージドロップ率を比較する。メッセージドロップ率は

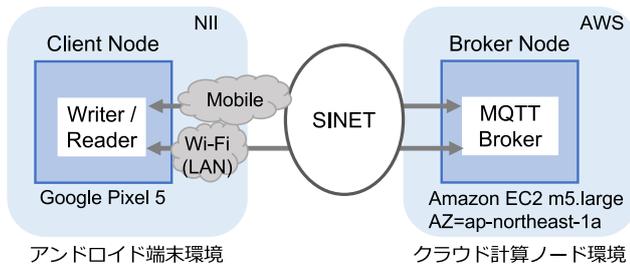


図 5 実験概要

Fig. 5 An outline of the experiment for evaluation.

メッセージ全体のうち再送が発生したメッセージの割合を示す。ここで、Writer と Reader の性能を別々に測定することが望ましいが、比較的通信遅延およびその変動が大きいモバイル環境で Writer, Reader, ブローカ間で厳密な時刻同期をとるのは困難である。Android 端末から送信される 100 B から 10 KB 程度の小規模メッセージ送信での通信スループット評価実験では、時刻同期の誤差の性能への影響も大きくなる。本実験結果は、Writer, Reader 単体で測定するより通信が劣ることが想定されるが、ベースライン性能、および通信・データの暗号化の影響を測定することを目的として、Writer と Reader は同じ端末で動作させることとした。また、測定プログラムでは指定されたメッセージサイズのデータを、指定された回数を上限として、Writer から送り続けるようにし、送信完了後 10 秒待ってもメッセージが Reader に届かなければ打ち切るようにした。これにより、通信状況によりメッセージの再送が発生する場合は、Reader で確認できるメッセージの到着数は上限値より少なくなる。実験では、100 B、1 KB のときの送信上限値を 20000、10 KB では 5000 とした。

表 1 に評価で用いた実験環境を示す。クライアントの Android 端末には Google Pixel 5、ブローカには AWS の m5.large インスタンスを用いた。また、モバイル環境ではモバイル SINET (LTE) と SINET L2VPN を介して AWS に接続され、Wi-Fi 環境では NII 所内の eduroam Wi-Fi 環境から SINET を介して AWS に接続される。クライアントには SINETStream Android v. 1.5.3、ブローカには Eclipse Mosquitto v1.6.2 を用いた。SINETStream のパラメータである consistency, tls, crypto は表 1 のように設定した。consistency の AT_LEAST_ONCE は、1 つ以上のメッセージが届くことを意味し、メッセージ送信に失敗した場合は再度メッセージを送信することになる。tls は TLS (通信暗号化)、crypto は AES を用いたデータ暗号化に関するパラメータである。

図 6 にモバイル環境と Wi-Fi 環境での TLS, AES の有無による通信スループットの比較結果を示す。各グラフの左側はモバイル環境、右側は Wi-Fi 環境の結果を表す。縦軸は通信スループット値を MB/s で、横軸の m/w-100/1K/10K の表記は m はモバイル、w は Wi-Fi の結

表 1 実験環境

Table 1 The experimental settings.

利用した計算機	
クライアント	Google Pixel 5 Qualcomm Snapdragon 765G, 8 コア 8GB メモリ, Android 11
ブローカ	Amazon EC2 m5.large VM, 2 スレッド Intel Xeon Platinum 8175 M 8GB メモリ, 10Gbps CentOS Linux release 7.7.1908
ソフトウェア	
SINETStream	SINETStream 1.5.3 for Android
MQTT	Mosquitto, mosquitto-1.6.2.tar.gz (source package)
SINETStream parameters	
consistency	AT_LEAST_ONCE
tls (TLS)	check_hostname: off
crypto (AES)	key_length: 256, mode: GCM

果を示し、-の後ろはメッセージサイズを bytes を表す。Plain, TLS, AES, TLS+AES は、Plain は TLS, AES なしの結果であり、それ以外は TLS, AES, およびその両方を適用した結果である。図 6 から、メッセージサイズの違いによるスループットの比較では、いずれの場合もメッセージサイズが大きくなるにつれ通信スループットが向上する。これは 1 つのメッセージ送信に関わる初期オーバーヘッドが相対的に小さくなるためである。また、モバイルと Wi-Fi の比較では、100 B、1 K ではほぼ同等のスループットであるのに対し、10 KB では Wi-Fi 環境のほうがスループットが高く、高性能であった。両環境における Plain, TLS, AES, TLS+AES の比較では、TLS や AES を適用しても大幅なスループットの低下はみられなかった。ただし、Plain より追加の処理を含む条件のほうがスループットが高くなるケースが見受けられる。これは、Android 端末での処理スループットよりモバイルおよび Wi-Fi 環境の通信スループットが十分になく、メッセージドロップ率が大きくなっていることが原因の 1 つと考えられる。ドロップ率については、図 8 の実験のところで改めて説明する。また、Wi-Fi 環境での TLS の通信スループットが高くなっているのは、上記に加えて通信環境の変化による影響の可能性も高いと考えられる。

図 7 では、モバイル環境と Wi-Fi 環境での Plain, TLS, AES, TLS+AES の通信遅延の最小値を示す。縦軸は通信遅延を ms で表し、横軸は、図 6 同様にモバイルまたは Wi-Fi とメッセージサイズの違いを表している。通信環境の変動と次に示すメッセージドロップの影響が大きいため、最小値で比較する。図 7 から、Wi-Fi 環境よりモバイル環境のほうが通信遅延が大きくなった。TLS および AES の有無の比較では、AES を適用しているもののほうが通信遅延が大きくなることが示された。

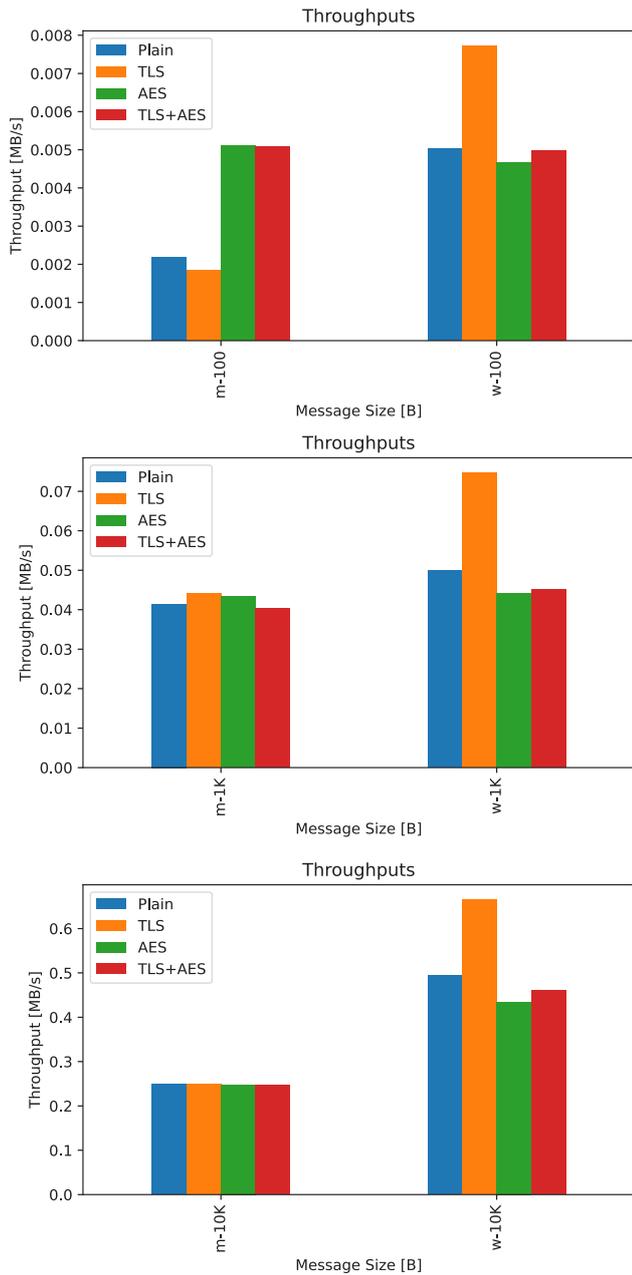


図6 モバイル環境とWi-Fi環境でのPlain, TLS, AES, TLS+AESの通信スループットの比較. 左からメッセージサイズ100B, 1KB, 10KBの結果を示す

Fig. 6 Comparison of the messaging throughputs in the mobile (left) and Wi-Fi (right) environments.

図8にモバイル環境とWi-Fi環境でのPlain, TLS, AES, TLS+AESのメッセージドロップ率を示す. 縦軸はドロップ率%, 横軸はモバイルまたはWi-Fiとメッセージサイズの違いを表す. メッセージドロップ率は, 総メッセージ数に対して再送が発生したメッセージの割合を示す. 図8から, モバイルおよびWi-Fi環境でメッセージサイズが100B, 1KBのときにはPlainのドロップ率がほぼ100%となっており, 送信レートに対して十分な通信レートが保てていないことが分かる. 同様に, TLSも高いドロップ率を示している. 一方, AESおよび

表2 モバイル環境でのメッセージ送信レート[messages/sec]の実測値

Table 2 The results of message send rate in the mobile environment.

条件\メッセージサイズ	100B	1KB	10KB
plain	20.2	42.5	25.5
TLS	18.9	45.2	25.5
AES	52.3	44.5	25.4
TLS+AES	51.9	41.3	25.3

TLS+AESではドロップ率は40%以下となっていた. 通常の測定であれば, Plain, TLS, AES, TLS+AESの順に性能が劣化することが予想されるが, モバイルおよびWi-Fi環境で通信環境が不安定であるのに加え, ドロップによって性能が大幅に劣化する. すなわち, 十分な通信帯域のない環境では送信レートを抑制しないと無駄な再送処理が大量に発生してしまい, 性能劣化と計算資源の浪費が発生すると考えられる. 既発表研究[3]において, Linux環境でJava版SINETStreamを用いて有線LAN環境とモバイル環境で同様にconsistencyをAT_LEAST_ONCEとした場合の測定を行い, 通信帯域の狭いモバイル環境のみドロップが大量に発生することを確認している. また, 表2に示すメッセージ送信レートの実測値では, メッセージサイズが100Bのときに処理遅延の大きいデータ暗号化処理を行うAESとTLS+AESの送信レートは50[messages/sec]以上であったのに対し, PlainとTLSの送信レートは約20[messages/sec]まで低下していた. 一方, 図8のメッセージドロップ率は送信レートに反比例して大きくなっており, 送信レートの調整でドロップ率と通信スループットの改善の余地があると考えられる.

4. SINETStream Android アプリの開発

応用研究分野の研究者に対してAndroid端末を用いたIoTアプリケーションシステムの構築を支援するため, SINETStream Androidのコアライブラリおよびヘルパーライブラリを用いたテキスト送受信アプリEchoとセンサ情報収集アプリSensorを開発した. また, 研究・教育用途での利用を支援するため, EchoとSensorを用いたチュートリアルも開発, 公開[10]した. SINETStream Android版および本研究で開発したアプリは, Android v8.0以降に対応している.

4.1 テキスト送受信アプリ Echo

Echoは, IoTシステムのデータ収集における基本的な振る舞いを確認することができるテキスト送受信アプリである. あらかじめ利用者がMQTTブローカの接続情報を設定しておき, Android端末のキーボードで入力した文字列をMQTTブローカに送信すると, 端末上に送信した

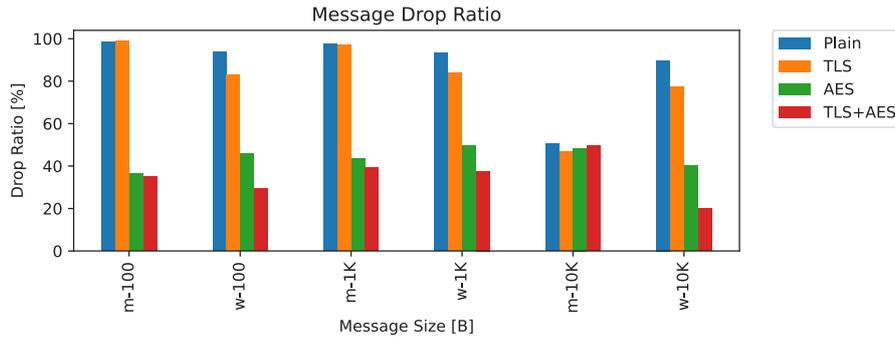


図8 モバイル環境と Wi-Fi 環境での Plain, TLS, AES, TLS+AES のメッセージドロップ率の比較
 Fig. 8 Comparison of the message drop ratios in the mobile and Wi-Fi environments.

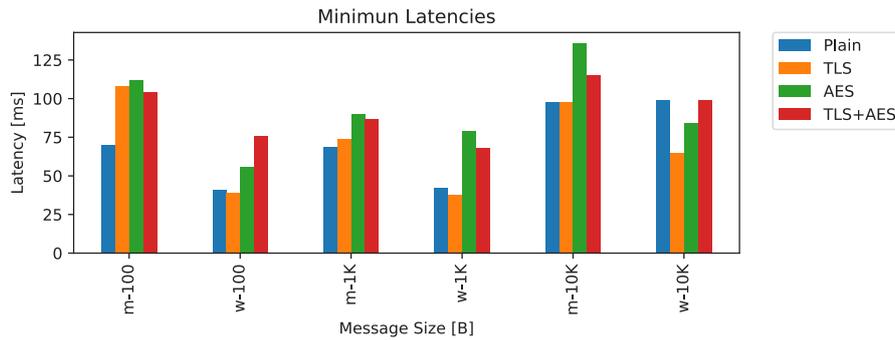


図7 モバイル環境と Wi-Fi 環境での Plain, TLS, AES, TLS+AES の通信遅延の最小値の比較
 Fig. 7 Comparison of the latencies in the mobile and Wi-Fi environments.



図9 Echo アプリの状態遷移
 Fig. 9 Transition of the Echo app.

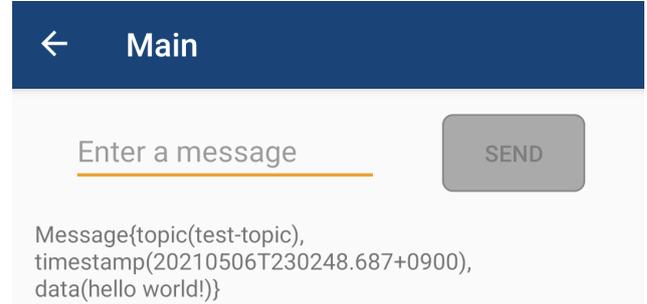


図10 Echo アプリの (a) 主画面上部のスナップショット. メッセージ受信結果
 Fig. 10 Echo main screen.

メッセージが表示される. これにより, IoT システムのデータ収集における基本的な振る舞いを確認することができる.

図9に Echo アプリの状態遷移を示す. 図中の (a) は Echo の主画面, (b) は初期画面, (c) は設定画面を表す. アプリを起動すると, まず (b) 初期画面に遷移する. (b) 初期画面からの遷移は以下のとおりである.

1. (b) 初期画面で右側の Settings ボタンを押下すると (c) 設定画面に遷移する.
2. (b) 初期画面で左側の Run ボタンを押下すると, (a) 主画面に遷移する.

(c) 設定画面では, SINETSStream のコンフィグファイル相当のパラメータ設定が可能である. 具体的には, トピック名やブローカのアドレス, ポート番号等を設定することができる. また, (a) 主画面ではソフトウェアキーボードからテキストを入力して SEND ボタンを押すと, 入力されたテキストを含むメッセージが MQTT ブローカに送信される. この際, メッセージは (c) 設定画面で指定されたトピック名で SINETSStream Android コアライブラリの Writer API を介して MQTT ブローカに送信される. Echo アプリは指定されたトピックの Reader でもあるため, 送信したメッセージを MQTT ブローカを介して受信し, その結果を (a) 主画面に表示する. 図10に (a) 主画

面の上部のスナップショットを示す。メッセージ受信結果では、入力したテキストデータだけでなくメッセージ全体が表示されている。メッセージには、トピック名、タイムスタンプ、入力したテキストデータが含まれている。

4.2 センサ情報収集アプリ Sensor

Sensor は、Android 端末に装備された多種センサの情報を収集するアプリであり、IoT アプリケーションシステムでそのまま利用可能となっている。あらかじめ利用者が MQTT ブローカへの接続情報およびセンサデータ送出間隔を設定しておき、利用するセンサタイプを選択すると、指定された送出間隔で対象となるセンサデータが MQTT ブローカに送信される。

Sensor アプリの状態遷移は図 9 の Echo アプリとほぼ同様であるが、d 設定画面と e 主画面での操作が異なっている。d 設定画面では、SINETStream のコンフィグファイル相当のパラメータの設定が可能であるほか、センサデータの送出間隔やユーザ定義情報等の Sensor アプリ固有のパラメータが設定できる。e 主画面では、送出するセンサのタイプを指定できる。図 11 は Sensor アプリの主画面であり、2 で SINETStream Android ヘルパーライブラリで認識されたセンサの一覧が表示されるため、利用したいセンサタイプを指定することができる。ただし、表示されるセンサの一覧は Android 端末の機種によって異なる。このほか、図 11 の 1 の星印ではヘルパーライブラリのセンササービスが稼働中であることを示している。3 の RUN/STOP ボタンでは、センサデータの取得および MQTT ブローカへのメッセージ送信の開始/停止を行う。4 ではセンサデータ送信中に送信時刻、総送信メッセージ数が表示され、5 のリセットボタンではメッセージ送信停止時に統計情報をリセットすることができる。

4.3 Sensor を用いたチュートリアル

研究・教育用途での利用を支援するため、Echo と Sen-



図 11 Sensor アプリの主画面

Fig. 11 The main screen of the Sensor app.

sor を用いたチュートリアルを開発し、IoT アプリケーションシステムの構築、利用を簡単に体験できるようにする。Echo アプリでは、Writer と Reader の機能がアプリで実装されており、Python/Java 版のチュートリアルで提供されているサーバ用 Docker コンテナイメージ [11] を用いて MQTT ブローカを用意すれば動作を確認することができる。一方、Sensor アプリでは Writer 側の機能しか実装されておらず、動作確認には Reader 側の機能を持つプログラムが必要となる。そこで、Sensor アプリが送信したセンサデータを受け取って蓄積、可視化までを行う Sensor チュートリアルサーバ用 Docker コンテナイメージも開発し、Android のセンサを活用した IoT アプリケーションシステムの構築、利用が体験できるようにした。本稿では、Sensor のチュートリアルについて紹介する。

Sensor チュートリアルの手順を以下に示す。

- (1) Sensor チュートリアル用サーバを設定する。
- (2) Android 端末でアプリをインストールし、実行する。
- (3) 観測用端末で可視化結果を確認する。

まず、(1) ではサーバ計算機上で Sensor チュートリアル用のサーバ Docker コンテナを実行する。(2) では、4.2 節の操作により、アプリのインストール、センサデータ送出間隔の設定、センサタイプの選択、センサデータの送信処理を行う。(3) では、観測用端末で以下の URL にアクセスすると、図 13 の画面でセンサデータが取得できていることが確認できる。

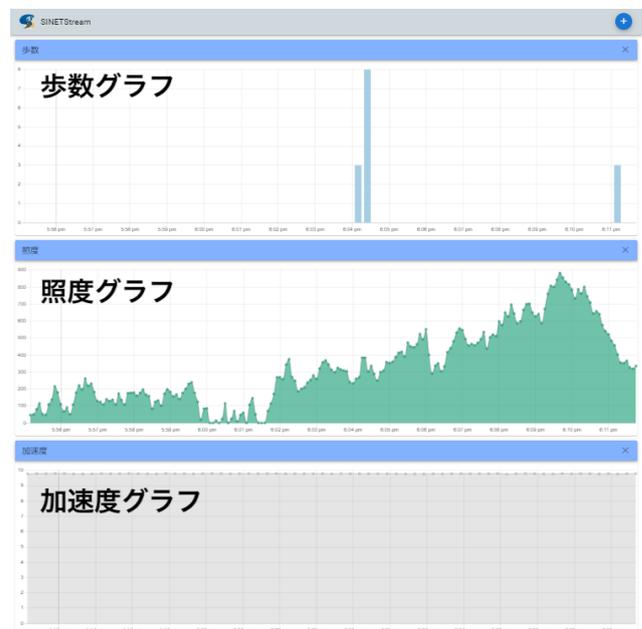


図 13 Sensor チュートリアルでのセンサデータ可視化画面のスナップショット。上から歩数、照度、加速度の時系列グラフが表示されている

Fig. 13 A snapshot of sensor data graphs used in the Sensor tutorial. The time series graphs show sensor data captured by step counter, light and acceleration sensors.

http://<server_address>/chart.html

Sensor チュートリアル用のサーバ Docker コンテナの詳細を図 12 に示す. 本コンテナでは, いずれもオープンソースソフトウェアである MQTT ブローカの Eclipse Mosquitto [12], メッセージブローカ Kafka, オブジェクトストレージ MinIO [13], Web サーバ nginx [14] がインストールされている. また, Kafka Connect 1, Kafka Connect 2 は, Kafka Connect と呼ばれるツールを表しており, Kafka と他のブローカーやストリーム処理系, ストレージ等との接続を容易にする機能を提供している. チュートリアル用のサーバコンテナでは, Kafka Connect 1 で Mosquitto に書き込まれたセンサデータを Kafka へ, Kafka Connect 2 で Kafka に書き込まれたセンサデータを MinIO へ蓄積するように設定した. MinIO は Amazon S3 (Simple Storage Service) 互換のオブジェクトストレージであり, nginx を介して指定されたセンサデータを取り出し, JavaScript で可視化できるようにした. 別途処理プログラムを用意して Mosquitto からストレージにデータを格納する方法も可能であるが, 本研究では Kafka ブローカを間に挟む構成とした. これは, Mosquitto では収集されたデータが永続化されないのに対して Kafka では一定期間永続化される点, Kafka Connect で処理プログラムを用意しなくても上記のような処理ができる点による.

図 13 は, 上から歩数, 照度, 加速度のセンサデータが時系列にグラフ化されたものである. 表示されるグラフは, 前述のセンサ 3 種だけでなく必要に応じて追加・削除することができる. Sensor アプリを実行すると, 定期的に指定したセンサのデータがグラフ上で更新されることを確認した. なお, 図 13 では加速度のグラフにあまり変化がみられない. 加速度センサデータは, Android 端末に対して直交座標系 xyz 軸方向にかかる加速度値 $[m/s^2]$ を -1 倍したもので表現されている. xy 軸が水平方向, z 軸が鉛直方向であり, 静止状態では $(0, 0, 9.8)$, 自由落下状態では $(0, 0, 0)$ となる. 本研究では動作確認用としてすべてのセンサデータ結果を 2 次元のグラフで表しているため, 可視化時に加速度センサデータを便宜的にベクトル長に変換している. ベクトル長は, 歩行程度の移動では 0.001 前後しか変動しないため, ほぼすべてのデータが

9.8 のところにプロットされた結果が得られている.

4.4 Sensor アプリを用いた実証実験

最後に, Sensor アプリを使ってセンサ情報が収集, 活用できることを示す. Sensor アプリを起動した Android 端末を東京海洋大学の練習船の船橋前方窓付近に設置し, 2022 年 2 月 15 日から 3 日間の実験航海時の様々なセンサ情報を 30 秒ごとに収集した. 船内に設置したため, 測定中は給電した状態でセンサ情報を収集した. Android 端末には Google Pixel 3a XL を用い, モバイル SINET へ接続するための SINET SIM を装着した. Sensor アプリからモバイル SINET, SINET L2VPN を介して NII の所内クラウド上に配備した Mosquitto MQTT サーバにデータを送信する. サーバサイドの構成は図 12 と同様であり, 送信されたデータは MQTT サーバから Kafka を介してオブジェクトストレージにデータを格納するようにした.

図 14 は, 航海中に Sensor アプリで取得したセンサ情報のうち Android 端末の GPS センサで取得した位置情報を可視化したものである. Sensor アプリから送信されたセンサデータのうち, 位置情報を可視化するプログラムを開発した. センサデータは JSON 形式でオブジェクトスト



図 14 Android センサ情報収集アプリの GPS の利用例. 国土地理院の提供する地理院タイル [15] を加工して作成

Fig. 14 An example of GPS data acquisition using the Sensor app. This figure is generated using the map provided at [15].

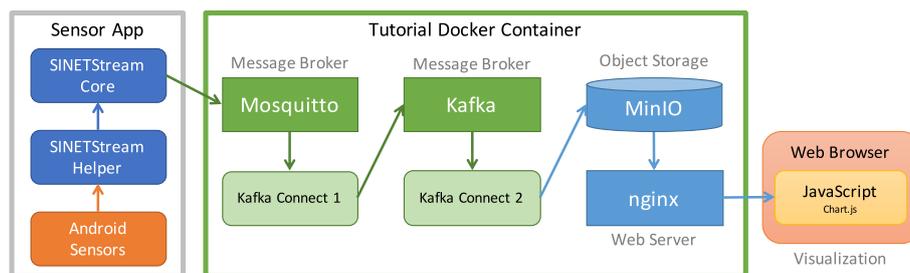


図 12 Sensor チュートリアルのサーバ用 Docker コンテナの概要

Fig. 12 A server-side Docker container for the Sensor tutorial.

レーズに保存されているため、容易に読み出し、処理することができた。なお、図 14 は国土地理院の提供する地理院タイル [15] を加工して作成した。図 14 の航路では通信断の発生しない安定した電波状況が維持されていたため、Sensor アプリを用いることで位置情報を含む 14 種類のセンサ情報を 3 日間分欠損することなく正常に収集できていることが確認できた。また、3.3 節で確認したようなメッセージドロップも発生していなかった。

5. 関連研究

Android スマートフォンを IoT アプリケーションシステムのセンサ端末として活用する試みは複数ある。文献 [16] では、患者の監視を目的として加速度センサ、温度センサ、心電図センサのデータを Bluetooth または NFC (Near Field Communication) 経由で Android 端末に収集して可視化するとともに、セントラルサーバに送信するマイクロコントローラを開発している。文献 [17] も eHealth を目的とし、患者に装着した心電図および心拍数のセンサを Wi-Fi 802.11 で Android 端末に接続し、GPS データとともに収集し、リアルタイムに可視化するシステムを構築している。これらのシステムでは、利用シナリオに応じて個別のセンサに特化したシステムが構築され、独自の方式でデータ収集等を行っているのに対し、本研究ではより広範な目的で利用可能なアプリとして開発されている。特に、Android システムが管理している複数センサに対応している点、サーバへのデータ収集は SINETStream により多種メッセージブローカに対応している点、SINETStream の持つ IoT システムに必要な各種機能が活用できる点で異なる。

Android 端末のセンサ情報をクラウドで収集・蓄積・解析し、その結果を活用する IoT システムの 1 つとして、自然災害による被害リスクを軽減することを目的とした早期警鐘システム [18] がある。このシステムでは、広域に分散している利用者が保有する多数の Android 端末で検知された揺れの情報をクラウドソーシングで収集し、クラウドで地震であると予測されると各 Android 端末の利用者に通知する。本研究で開発した Sensor アプリもこのような用途で活用されることが期待できる。

6. まとめと今後の課題

本研究では、多種センサがあらかじめ搭載された Android 端末を活用した研究・教育用 IoT アプリケーションシステムの構築支援のため、SINETStream Android 版ライブラリを開発し、その基本性能を示した。また、本ライブラリのコアライブラリと、Android 端末のセンサ情報収集を支援するヘルパーライブラリを用いて、Android 用のテキスト送受信アプリ SINETStream Android Echo (Echo) とセンサ情報収集アプリ SINETStream Android

Sensor Publisher (Sensor) を開発した。Echo アプリを用いることで、IoT システムのデータ収集における基本的な振る舞いを確認することが可能である。Sensor アプリは Android 端末をセンサとする IoT アプリケーションシステムでの利用が可能であり、IoT を活用した研究開発を加速させることができると期待している。また、研究・教育用途での利用を支援するため、Echo と Sensor を用いたチュートリアルも開発、公開した。Sensor チュートリアルでは、センサデータの収集、蓄積、可視化を可能にするサーバ用 Docker コンテナも開発し、本チュートリアルにより Android 端末を用いた IoT アプリケーションシステムの構築・利用が体験できるようにした。また、広域での実証実験から Sensor アプリが 3 日間の実験航海でのセンサ情報収集に活用できることを示した。

今後は、送信レートの調整による性能・利用効率の向上を図るとともに、スマートフォンで課題となるバッテリーの管理や、セキュリティ機能、メトリクス収集機能など、SINETStream Android 版の機能を拡充していく。また、Sensor アプリの利便性をより高めるとともに、実アプリケーションシステムへの適用を行っていく。さらに、大学等でのチュートリアル教材の授業での活用や Sensor アプリの IoT システムでの利用など、活用事例を共有していく。

本研究で開発したソフトウェア一式は、SINETStream のウェブサイト <https://sinetstream.net/> で公開している。SINETStream は、オープンソースソフトウェアであり、SINET 以外の環境でも利用可能である。

謝辞 本研究にご協力いただいた東京海洋大学の島浩太先生、総合研究大学院大学の陳明康様、数理技研の遠藤雅彦様、小泉敦延様、鯉江英隆様に深く感謝いたします。

本研究は、JST、CREST、JPMJCR21M3 の支援を受けたものです。

参考文献

- [1] 国立情報学研究所：モバイル SINET 実証実験、[\(https://www.sinet.ad.jp/wadci/\)](https://www.sinet.ad.jp/wadci/) (参照 2022-05-06)。
- [2] 国立情報学研究所：SINETStream、[\(https://www.sinetstream.net/\)](https://www.sinetstream.net/) (参照 2022-05-06)。
- [3] Takefusa, A., Sun, J., Fujiwara, I., Yoshida, H., Aida, K. and Pu, C.: SINETStream: Enabling Research IoT Applications with Portability, Security and Performance Requirements, *Proc. IEEE COMPSAC 2021*, pp.482-492 (2021).
- [4] 孫 静涛, 竹房あつ子, 藤原一毅, 吉田 浩, 合田憲人：IoT アプリ構築支援のための SINETStream Android プラグインの開発, マルチメディア, 分散, 協調とモバイル (DICOMO2020) シンポジウム論文集, pp.421-427(2020).
- [5] 竹房あつ子, 小林久美子, 北川直哉, 孫 静涛, 吉田浩, 合田憲人：IoT アプリケーションのための SINETStream ベース Android センサ情報収集アプリの開発, マルチメディア, 分散, 協調とモバイル (DICO-

- MO2021) シンポジウム論文集, pp.1068-1074 (2021).
- [6] MQTT: MQTT: The Standard for IoT Messaging, <http://mqtt.org/> (参照 2022-05-06).
- [7] Apache: Apache Kafka, <https://kafka.apache.org/> (参照 2022-05-06).
- [8] Kreps, J., Narkhede, N. and Rao, J.: Kafka: a Distributed Messaging System for Log Processing, *NetDB workshop 2011*, pp.1-5 (2011).
- [9] Eclipse Foundation: Eclipse Paho, <https://www.eclipse.org/paho/> (参照 2022-05-06).
- [10] 国立情報学研究所: SINETStream Android 版チュートリアル, <https://www.sinetstream.net/docs/tutorial-android/> (参照 2022-05-06).
- [11] 国立情報学研究所: SINETStream Python/Java 版チュートリアル, <https://www.sinetstream.net/docs/tutorial/> (参照 2022-05-06).
- [12] Eclipse Foundation: Eclipse Mosquitto, <https://mosquitto.org/> (参照 2022-05-06).
- [13] MinIO: Multi-Cloud Object Storage, <https://min.io/> (参照 2022-05-06).
- [14] nginx: <http://nginx.org/> (参照 2022-05-06).
- [15] 国土地理院: 地理院タイル, <https://maps.gsi.go.jp/development/ichiran.html> (参照 2022-05-06).
- [16] Yi, W.-J., Jia, W. and Saniie, J.: Mobile Sensor Data Collector using Android Smartphone, *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp.956-959 (online), DOI: 10.1109/MWSCAS.2012.6292180 (2012).
- [17] Khalaf, A. and Abdoola, R.: Wireless Body Sensor Network and ECG Android Application for eHealth, *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*, pp.1-4 (online), DOI: 10.1109/ICABME.2017.8167526 (2017).
- [18] Heryana, A., Nugraheni, E., Kusumo, B., Rojje, A. F. and Setiadi, B.: Applying Agile Methods in Designing an Earthquake and Landslide Early Warning System Application for Android, *2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pp.80-84 (online), DOI: 10.1109/IC3INA.2017.8251744 (2017).



竹房 あつ子 (正会員)

2000年博士(理学)(お茶の水女子大学)取得。日本学術振興会特別研究員(DC2, PD), お茶の水女子大学大学院助手, 産業技術総合研究所研究員, 主任研究員を経て2016年国立情報学研究所アーキテクチャ科学研究系准教授, 2021年同教授, 現在に至る。総合研究大学院大学複合科学研究科情報学専攻教授兼任。並列分散処理, グリッド, クラウド, エッジ, IoTに関する研究に従事。ACM, IEEE, 電子情報通信学会各会員。本会シニア会員。



小林 久美子 (正会員)

1992年電気通信大学電気通信学部電子情報学科卒業。同年日本無線株式会社入社。2012年電気通信大学大学院情報システム学研究科博士後期課程単位取得済み退学。2014年博士(工学)取得。2016年より国立情報学研究所クラウド支援室, 2020年同クラウド基盤研究開発センター, 現在に至る。クラウド, IoT関連の研究・開発および大学・研究機関のクラウド導入支援に従事。IEEE, 電子情報通信学会各会員。



北川 直哉 (正会員)

2014年3月名古屋大学大学院情報科学研究科博士後期課程修了・博士(情報科学)。同年4月より同大学情報基盤センター研究員を経て, 同年10月より東京農工大学大学院工学研究院先端情報科学部門助教。2020年4月より国立情報学研究所学術ネットワーク研究開発センター特任准教授, 現在に至る。ネットワークシステム, 情報セキュリティに関する研究に従事。IEEE, 情報処理学会各会員。



孫 静涛 (正会員)

2013年東京工科大学大学院バイオ情報メディア研究科コンピュータサイエンス専攻博士課程前期修了。2016年総合研究大学院大学複合科学研究系情報学専攻博士課程後期修了。博士(情報学)。同年国立情報学研究所アーキテクチャ科学研究系特任研究員。2021年日立製作所 研究開発グループ デジタルテクノロジーイノベーションセンター サービスコンピューティング研究部シニア研究員を経て, 現在に至る。エコノミデータセンタ, クラウドコンピューティング, IoTの研究に従事。ACM, IEEE, CCF各会員。



吉田 浩 (正会員)

1978年東京大学工学部電子工学科卒業。1980年同大学大学院工学系研究科修士課程修了。富士通株式会社において, ソフトウェア, ストレージシステム, パブリッククラウドサービスの企画・開発に従事。2015年より国立情報学研究所クラウド基盤研究開発センターにおいて, クラウド関連の研究・開発および大学・研究機関のクラウド導入支援に従事。IEEE 会員。



合田 憲人 (正会員)

1997年博士(工学)(早稲田大学)取得。1992年早稲田大学情報科学研究教育センター助手, 1997年東京工業大学大学院情報理工学研究科数理・計算科学専攻助手, 1999年同大学院総合理工学研究科知能システム科学専攻講師, 2003年同研究科物理情報システム専攻助教授, 2007年国立情報学研究所特任教授, 2015年同教授, 現在に至る。科学技術振興機構さきがけ研究員(2001年~2005年), ハワイ大学 Information and Computer Sciences Department 客員研究員(2007年), 東京工業大学大学院総合理工学研究科物理情報システム専攻連携教授(2007年~2016年), 総合研究大学院大学複合科学研究科情報学専攻教授(2008年~現在)等を兼任。電子情報通信学会, IEEE, ACM 各会員。